Logic: Proof procedures, soundness and correctness

CPSC 322 – Logic 2

Textbook  5.2

March 7, 2011
Lecture Overview

Recap: Propositional Definite Clause Logic (PDCL)
- Syntax
- Semantics

- More on PDCL Semantics

- Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    - Pseudocode and example
    - Time-permitting: Soundness
    - Time-permitting: Completeness
Course Overview

Environment
- Deterministic
  - Arc Consistency
  - Variable Elimination
- Stochastic
  - Bayesian Networks
    - Variable Elimination
  - Decision Networks
    - Variable Elimination
    - Value Iteration
  - Markov Processes
    - Value Iteration

Problem Type
- Static
  - Logic
    - STRIPS
    - Search
    - As CSP (using arc consistency)
    - Static problems, but with richer representation
  - Constraint Satisfaction
    - Variables + Constraints
    - Search
  - Representation
  - Reasoning
  - Technique
- Sequential
  - Planning
  - Decision Theory
**Definition (RRS)**

A Representation and Reasoning System (RRS) consists of:

- **syntax**: specifies the symbols used, and how they can be combined to form legal sentences
- **semantics**: specifies the meaning of the symbols
- **reasoning theory or proof procedure**: a (possibly nondeterministic) specification of how an answer can be produced.

Propositional definite clause logic (PDCL) is one such Representation and Reasoning System
Example: Electrical Circuit

\[\begin{align*}
\text{light}_1 &\leftarrow \text{live}_w_0, \\
\text{live}_w_0 &\leftarrow \text{live}_w_1 \land \text{up}_s_2, \\
\text{live}_w_1 &\leftarrow \text{live}_w_2 \land \text{down}_s_2, \\
\text{live}_w_2 &\leftarrow \text{live}_w_3 \land \text{up}_s_1, \\
\text{live}_w_3 &\leftarrow \text{live}_w_4, \\
\text{live}_w_4 &\leftarrow \text{live}_w_3 \land \text{up}_s_3, \\
\text{live}_p_1 &\leftarrow \text{live}_w_3, \\
\text{live}_w_3 &\leftarrow \text{live}_w_5 \land \text{ok}_cb_1, \\
\text{live}_w_5 &\leftarrow \text{live}_w_6, \\
\text{live}_w_6 &\leftarrow \text{live}_w_5 \land \text{ok}_cb_2, \\
\text{live}_w_5 &\leftarrow \text{live}_outside, \\
\text{lit}_l_1 &\leftarrow \text{light}_l_1 \land \text{live}_l_1 \land \text{ok}_l_1, \\
\text{lit}_l_2 &\leftarrow \text{light}_l_2 \land \text{live}_l_2 \land \text{ok}_l_2.
\end{align*}\]
Propositional Definite Clauses: Syntax

Definition (atom)  
An **atom** is a symbol starting with a lower case letter.  
Examples: p₁, live_l₁

Definition (body)  
A **body** is an atom or is of the form \( b₁ \land b₂ \) where \( b₁ \) and \( b₂ \) are bodies.  
Examples: p, ok_w₁ ∧ live_w₀, p₁ ∧ p₂ ∧ p₃ ∧ p₄

Definition (definite clause)  
A **definite clause** is an atom or is a rule of the form \( h \leftarrow b \) where \( h \) is an atom ("head") and \( b \) is a body. (Read this as `h if b`.)

Examples: p₁ ← p₂ ∧ p₃ ∧ p₄, live_w₀ ← live_w₁ ∧ up_s₂

Definition (KB)  
A **knowledge base (KB)** is a set of definite clauses  
Example: \{p₂, p₃, p₄, p₁ ← p₂ ∧ p₃ ∧ p₄, live_l₁\}
atoms

\[
\begin{align*}
\text{light}_1. \\
\text{light}_2. \\
\text{ok}_1. \\
\text{ok}_2. \\
\text{ok}_{\text{cb}1}. \\
\text{ok}_{\text{cb}2}. \\
\text{live}_{\text{outside}}. \\
\end{align*}
\]

definite clauses, KB

rules

\[
\begin{align*}
\text{live}_1 & \leftarrow \text{live}_w0. \\
\text{live}_w0 & \leftarrow \text{live}_w1 \land \text{up}_s2. \\
\text{live}_w0 & \leftarrow \text{live}_w2 \land \text{down}_s2. \\
\text{live}_w1 & \leftarrow \text{live}_w3 \land \text{up}_s1. \\
\text{live}_w2 & \leftarrow \text{live}_w3 \land \text{down}_s1. \\
\text{live}_l2 & \leftarrow \text{live}_w4. \\
\text{live}_w4 & \leftarrow \text{live}_w3 \land \text{up}_s3. \\
\text{live}_p1 & \leftarrow \text{live}_w3. \\
\text{live}_w3 & \leftarrow \text{live}_w5 \land \text{ok}_{\text{cb}1}. \\
\text{live}_p2 & \leftarrow \text{live}_w6. \\
\text{live}_w6 & \leftarrow \text{live}_w5 \land \text{ok}_{\text{cb}2}. \\
\text{live}_w5 & \leftarrow \text{live}_{\text{outside}}. \\
\text{lit}_l1 & \leftarrow \text{light}_1 \land \text{live}_1 \land \text{ok}_1. \\
\text{lit}_l2 & \leftarrow \text{light}_2 \land \text{live}_2 \land \text{ok}_2.
\end{align*}
\]
Lecture Overview

• Recap: Propositional Definite Clause Logic (PDCL)
  - Syntax
  - Semantics
• More on PDCL Semantics
• Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    • Pseudocode and example
    • Time-permitting: Soundness
    • Time-permitting: Completeness
Propositional Definite Clauses: Semantics

Semantics allows you to relate the symbols in the logic to the domain you're trying to model.

**Definition (interpretation)**
An interpretation \( I \) assigns a truth value to each atom.

**Definition (truth values of statements)**
- A **body** \( b_1 \land b_2 \) is true in \( I \) if and only if \( b_1 \) is true in \( I \) and \( b_2 \) is true in \( I \).
- A **rule** \( h \leftarrow b \) is false in \( I \) if and only if \( b \) is true in \( I \) and \( h \) is false in \( I \).
PDC Semantics: Example

Truth values under different interpretations
F=false, T=true

<table>
<thead>
<tr>
<th></th>
<th>a₁</th>
<th>a₂</th>
<th>a₁ ∧ a₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I₂</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>I₃</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I₄</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>h</th>
<th>b</th>
<th>¬b</th>
<th>¬b ⊃ h</th>
<th>h ← b</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>I₂</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I₃</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>I₄</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

h ← b (“h if b”) is only false if b is true and h is false
**PDC Semantics: Example for models**

**Definition (model)**

A *model* of a knowledge base KB is an interpretation in which every clause in KB is true.

```
KB =
    \[
    \begin{cases}
    p \leftarrow q \\
    q \\
    r \leftarrow s
    \end{cases}
    \]
```

Which of the interpretations below are models of KB?

<table>
<thead>
<tr>
<th>I_1</th>
<th>I_2</th>
<th>I_3</th>
<th>I_4</th>
<th>I_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
<th>p \leftarrow q</th>
<th>q</th>
<th>r \leftarrow s</th>
<th>Model of KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>yes</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>no</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>yes</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>yes</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model of KB</th>
<th>yes</th>
<th>no</th>
</tr>
</thead>
<tbody>
<tr>
<td>I_1</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>I_2</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>I_3</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>I_4</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>I_5</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
PDC Semantics: Example for models

Definition (model)
A model of a knowledge base KB is an interpretation in which every clause in KB is true.

\[
KB = \begin{cases}
p \leftarrow q \\
q \\
r \leftarrow s
\end{cases}
\]

Which of the interpretations below are models of KB?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>p ← q</th>
<th>q</th>
<th>r ← s</th>
<th>Model of KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>yes</td>
</tr>
<tr>
<td>I₂</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>I₃</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>I₄</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>I₅</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
Lecture Overview

• Recap: Propositional Definite Clause Logic (PDCL)
  - Syntax
  - Semantics

  More on PDCL Semantics

• Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    • Pseudocode and example
    • Time-permitting: Soundness
    • Time-permitting: Completeness
**PDCL Semantics: Logical Consequence**

**Definition (model)**  
A **model** of a knowledge base KB is an interpretation in which every clause in KB is true.

**Definition (logical consequence)**  
If KB is a set of clauses and g is a conjunction of atoms, G is a logical consequence of KB, written $KB \models g$, if g is true in every model of KB.

- We also say that g logically follows from KB, or that KB entails g.
- In other words, $KB \not\models g$ if there is no interpretation in which KB is true and g is false.
Definition (model)
A model of a knowledge base KB is an interpretation in which every clause in KB is true.

Definition (logical consequence)
If KB is a set of clauses and g is a conjunction of atoms, G is a logical consequence of KB, written $KB \models g$, if g is true in every model of KB

$KB = \begin{cases} 
  p \leftarrow q \\
  q \\
  r \leftarrow s 
\end{cases}$

Which of the following are true?

- $KB \not\models p$
- $KB \not\models q$
- $KB \models r$
- $KB \models s$
PDCL Semantics: Logical Consequence

**Definition (model)**
A **model** of a knowledge base KB is an interpretation in which every clause in KB is true.

**Definition (logical consequence)**
If KB is a set of clauses and g is a conjunction of atoms, G is a **logical consequence** of KB, written $KB \vDash g$, if g is true in every model of KB.

$KB = \begin{cases} 
  p \leftarrow q & \text{If KB is true, then q is true. Thus } KB \vDash q. \\
  q & \text{If KB is true then both } q \text{ and } p \leftarrow q \text{ are true, so } p \text{ is true ("p if q"). Thus } KB \vDash p. \\
  r \leftarrow s & \text{There is a model where } r \text{ is false, likewise for } s \text{ (but there is no model where } s \text{ is true and } r \text{ is false).}
\end{cases}$
Motivation for Proof Procedure

• We want a proof procedure that can find all and only the logical consequences of a knowledge base

• Why?
User’s View of Semantics

1. Choose a task domain: intended interpretation.
2. Associate an atom with each proposition you want to represent.
3. Tell the system clauses that are true in the intended interpretation: axiomatizing the domain.
4. Ask questions about the intended interpretation.
   – If $\text{KB \models g}$, then g must be true in the intended interpretation.
   – The user can interpret the answer using their intended interpretation of the symbols.
Computer’s view of semantics

• The computer doesn't have access to the intended interpretation.
  – All it knows is the knowledge base.

• The computer can determine if a formula is a logical consequence of KB.
  – If KB $\vDash g$ then g must be true in the intended interpretation.
  – Otherwise, there is a model of KB in which g is false.
    This could be the intended interpretation.
Role of semantics

In user's mind:
• l2_broken: light l2 is broken
• sw3_up: switch is up
• power: there is power in the building
• unlit_l2: light l2 isn't lit
• lit_l1: light l1 is lit

In computer:
• l2_broken ← sw3_up ∧ power ∧ unlit_l2.
• sw3_up.
• power ← lit_l1.
• unlit_l2.
• lit_l1.

Conclusion: l2_broken
- The computer doesn’t know the meaning of the symbols
- The user can interpret the symbols using their meaning
Lecture Overview

• Recap: Propositional Definite Clause Logic (PDCL)
  - Syntax
  - Semantics

• More on PDCL Semantics

Proof procedures

  - Soundness, Completeness, example
  - Bottom-up proof procedure
    • Pseudocode and example
    • Time-permitting: Soundness
    • Time-permitting: Completeness
Proofs

• A proof is a mechanically derivable demonstration that a formula logically follows from a knowledge base.

• Given a proof procedure $P$, \( KB \vdash_P g \) means $g$ can be derived from knowledge base $KB$ with the proof procedure.

• Recall $KB \models g$ means $g$ is true in all models of $KB$.

• Example: simple proof procedure $S$
  – Enumerate all interpretations
  – For each interpretation $I$, check whether all clauses in $KB$ hold
    • If all clauses are true, $I$ is a model
    • $KB \vdash_S g$ if $g$ holds in all such models
Definition (soundness)

A proof procedure P is **sound** if \( \text{KB} \vdash_P g \) implies \( \text{KB} \models g \).

**sound**: everything it derives follows logically from \( \text{KB} \) (i.e. is true in every model)

• Soundness of some proof procedure P: need to prove that

  **If** g can be derived by the procedure (\( \text{KB} \vdash_P g \))

  **then** g is true in all models of \( \text{KB} \) (\( \text{KB} \models g \))

• Example: simple proof procedure S
  
  – For each interpretation I, check whether all clauses in KB hold
    
    • If all clauses are true, I is a model
    
    • KB \( \vdash_S g \) if g holds in all such models

• The **simple proof procedure S is sound**:

  **If** KB \( \vdash_S g \), then it is true in all models, i.e. KB \( \models g \)
Completeness of a proof procedure

Definition (completeness)
A proof procedure $P$ is **complete** if $\text{KB} \not\models g$ implies $\text{KB} \vdash_P g$.

complete: everything that logically follows from KB is derived

- Completeness of some proof procedure $P$: need to prove that
  
  **If** $g$ is true in all models of $\text{KB}$ ($\text{KB} \not\models g$)
  **then** $g$ is derived by the procedure ($\text{KB} \vdash_P g$)

- Example: simple proof procedure $S$
  - For each interpretation $I$, check whether all clauses in KB hold
    - If all clauses are true, $I$ is a model
    - $\text{KB} \vdash_S g$ if $g$ holds in all such models

- The **simple proof procedure $S$ is complete**:
  **If** $\text{KB} \not\models g$, i.e. $g$ is true in all models, **then** $\text{KB} \vdash_S g$
Another example for a proof procedure

• Unsound proof procedure U:
  – U derives every atom: for any g, KB ⊢ U g

• Proof procedure U is complete:
  If KB ⊬ g, then KB ⊩ S g (because KB ⊢ U g for any g)

• Proof procedure U is not sound:
  Proof by counterexample: KB = {a ← b.}
  KB ⊢ U a, but not KB ⊬ a
  (a is false in some model, e.g. a=false, b=false)
Problem of the simplistic proof procedure

- Simple proof procedure: enumerate all interpretations
  - For each interpretation, check whether all clauses in KB hold
    - If all clauses hold, the interpretation is a model
    - $KB \vdash g$ if $g$ holds in all such models

- What’s the problem with this approach?
  - Space complexity
  - Time complexity
  - Not sound
  - Not complete
Problem of the simplistic proof procedure

•Enumerate all interpretations
  – For each interpretation, check whether all clauses of the knowledge base hold
  – If all clauses hold, the interpretation is a model

•Very much like the generate-and-test approach for CSPs

•Sound and complete, but there are a lot of interpretations
  – For n propositions, there are $2^n$ interpretations
Lecture Overview

• Recap: Propositional Definite Clause Logic (PDCL)
  - Syntax
  - Semantics
• More on PDCL Semantics
• Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    • Pseudocode and example
    • Time-permitting: Soundness
    • Time-permitting: Completeness
One rule of derivation, a generalized form of modus ponens:

- If “h ← b₁ ∧ … ∧ bₘ” is a clause in the knowledge base, and each bᵢ has been derived, then h can be derived.

This rule also covers the case when m = 0.
Bottom-up proof procedure

\[
C := \{\};
\]

repeat

\textbf{select} clause \( h \leftarrow b_1 \land \ldots \land b_m \) in \( KB \) such that \( b_i \in C \) for all \( i \), and \( h \notin C \);

\[
C := C \cup \{h\}
\]

until no more clauses can be selected.

\( KB \vdash g \) if \( g \in C \) at the end of this procedure.
C := {}; repeat
select clause h ← b₁ ∧ … ∧ bₘ in KB
such that bᵢ ∈ C for all i, and h ∉ C;
C := C ∪ {h}
until no more clauses can be selected.

a ← b ∧ c
a ← e ∧ f
b ← f ∧ k
c ← e
d ← k
e.
f ← j ∧ e
f ← c
j ← c
Bottom-up proof procedure: example

C := {}; 
repeat

select clause h ← b₁ ∧ ... ∧ bₘ in KB
such that bᵢ ∈ C for all i, and h ∉ C;

C := C ∪ {h}

until no more clauses can be selected.

a ← b ∧ c
a ← e ∧ f
b ← f ∧ k

{c,e}
{c,e,f}
{c,e,f,j}
{a,c,e,f,j}

Done.
Lecture Overview

• Recap: Propositional Definite Clause Logic (PDCL)
  - Syntax
  - Semantics
• More on PDCL Semantics
• Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    • Pseudocode and example
  Time-permitting: Soundness
  • Time-permitting: Completeness
Soundness of bottom-up proof procedure BU

**Definition (soundness)**
A proof procedure P is **sound** if KB ⊢ₚ g implies KB ⊨ g.

`sound`: everything it derives follows logically from KB (i.e. is true in every model)

```
C := {}; repeat
  select clause h ← b₁ ∧ … ∧ bₘ in KB such that bᵢ ∈ C for all i, and h ∉ C;
  C := C ∪ {h}
until no more clauses can be selected.
```

For **soundness of bottom-up proof procedure BU**: prove

**If** g ∈ C **at the end of BU procedure,**
**then** g **is true in all models of KB** (KB ⊨ g)
Soundness of bottom-up proof procedure BU

\[
C := \{\}; \\
\text{repeat} \\
\quad \text{select clause } h \leftarrow b_1 \land \ldots \land b_m \text{ in } KB \\
\quad \text{such that } b_i \in C \text{ for all } i, \text{ and } h \not\in C; \\
\quad C := C \cup \{h\} \\
\text{until} \text{ no more clauses can be selected.}
\]

For soundness of bottom-up proof procedure BU: prove

**If** \( g \in C \) at the end of BU procedure, **then** \( g \) is true in all models of KB (KB \( \models g \))

By contradiction: Suppose there is a \( g \) such that KB \( \vdash g \) but not KB \( \not\models g \).

- Let \( h \) be first atom added to \( C \) that's not true in every model of KB.
  - In particular, suppose \( I \) is a model of KB in which \( h \) isn't true.
- There must be a clause in KB of form \( h \leftarrow b_1 \land \ldots \land b_m \)
- Each \( b_i \) is true in \( I \). \( h \) is false in \( I \). So this clause is false in \( I \).
- Thus, \( I \) is not a model of KB. Contradiction: thus no such \( g \) exists.
Lecture Overview

- Recap: Propositional Definite Clause Logic (PDCL)
  - Syntax
  - Semantics
- More on PDCL Semantics
- Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    - Pseudocode and example
    - Time-permitting: Soundness
  - Time-permitting: Completeness
Minimal Model

- Observe that the C generated at the end of the bottom-up algorithm is a fixed point
  - Further applications of our rule of derivation will not change C!

**Definition (minimal model)**
The *minimal model* MM is the interpretation in which every element of BU’s fixed point C is true and every other atom is false.

- **Lemma:** MM is a model of KB.
  - Proof by contradiction. Assume that MM is not a model of KB.
    - Then there must exist some clause of the form \( h \leftarrow b_1 \land \ldots \land b_m \) in KB (with \( m \geq 0 \)) which is false in MM.
    - This can only occur when \( h \) is false and each \( b_i \) is true in MM.
    - Since each \( b_i \) belonged to C, we would have added \( h \) to C as well.
    - But MM is a fixed point, so nothing else gets added. Contradiction!
Completeness of bottom-up procedure

Definition (completeness)
A proof procedure is complete if $\text{KB} \not\models g$ implies $\text{KB} \vdash g$.

complete: everything that logically follows from KB is derived

For completeness of BU, we need to prove:

If $g$ is true in all models of $\text{KB}$ ($\text{KB} \not\models g$) then $g$ is derived by the BU procedure ($\text{KB} \vdash_{\text{BU}} g$)

Direct proof based on Lemma about minimal model:

• Suppose $\text{KB} \not\models g$. Then $g$ is true in all models of $\text{KB}$.
• Thus $g$ is true in the minimal model.
• Thus $g$ is generated by the bottom up algorithm.
• Thus $\text{KB} \vdash_{\text{BU}} g$. 
• PDCL syntax & semantics
  - Verify whether a logical statement belongs to the language of propositional definite clauses
  - Verify whether an interpretation is a model of a PDCL KB.
  - Verify when a conjunction of atoms is a logical consequence of a knowledge bases

• Bottom-up proof procedure
  • Define/read/write/trace/debug the Bottom Up (BU) proof procedure
  • Prove that the BU proof procedure is sound and complete